ESCUELA POLITECNICA NACIONAL

Materia: Métodos Numéricos

Nombre: Stiv Quishpe

link repositorio

https://github.com/stiv001/Tarea12.git

ODE Método de Euler

CONJUNTO DE EJERCICIOS

3. Utilice el método de Euler para aproximar las soluciones para cada uno de los siguientes problemas de valor inicial.

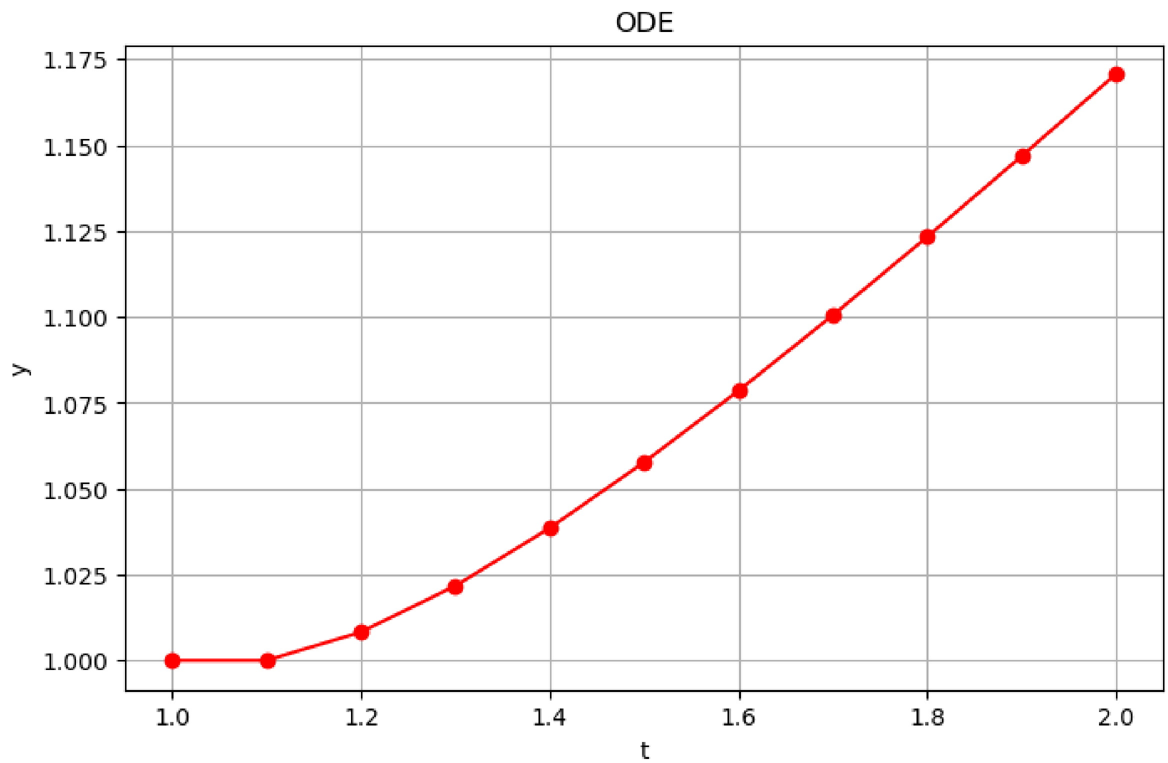
a. $y' = y/t - (y/t)^2, 1 \leq t \leq 2, y(1) = 1$, con $h = 0.1$

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
```

```
def euler_method(f, t0, y0, h, t_end):
    n = int((t_end - t0) / h) + 1
    t_values = np.linspace(t0, t_end, n)
    y_values = np.zeros(n)
    y_values[0] = y0

    for i in range(1, n):
        y_values[i] = y_values[i-1] + h * f(t_values[i-1], y_values[i-1])

    return t_values, y_values
```

```python
f_a = lambda t, y: y / t - (y / t)**2
t_a, y_a = euler_method(f_a, 1, 1, 0.1, 2)
results_a = pd.DataFrame({'t': t_a, 'y': y_a})

print("Resultados del Método de Euler:")
print(results_a)

plt.figure(figsize=(8, 5))
plt.plot(t_a, y_a, marker='o', linestyle='-', color='red')
plt.xlabel("t")
plt.ylabel("y")
plt.title("ODE")
plt.grid(True)
plt.show()
```

```
Resultados del Método de Euler:
      t         y
0   1.0  1.000000
1   1.1  1.000000
2   1.2  1.008264
3   1.3  1.021689
4   1.4  1.038515
5   1.5  1.057668
6   1.6  1.078461
7   1.7  1.100432
8   1.8  1.123262
9   1.9  1.146724
10  2.0  1.170652
```

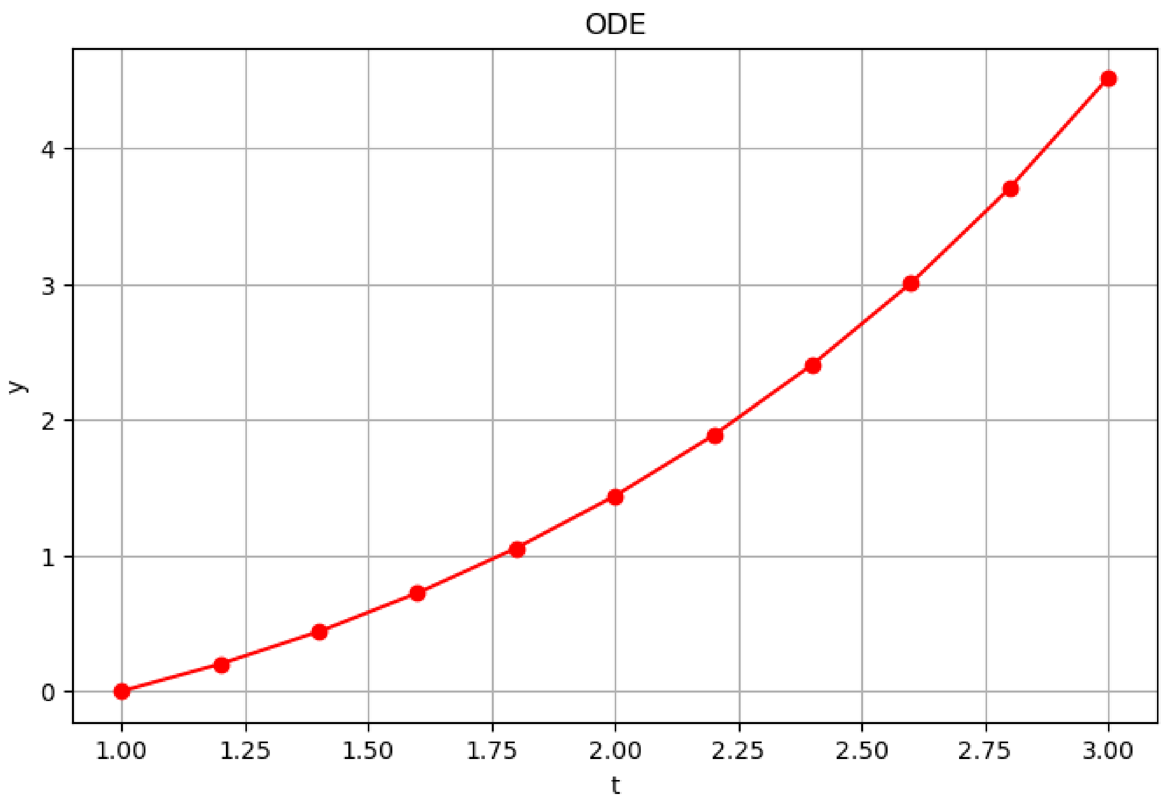


b. $y\acute{} = 1 + y/t + (y/t)^2, 1 \leq t \leq 3, y(1) = 0$, con $h = 0.2$

```
f_b = lambda t, y: 1 + y / t + (y / t)**2
t_b, y_b = euler_method(f_b, 1, 0, 0.2, 3)
results_b = pd.DataFrame({'t': t_b, 'y': y_b})
print("Resultados del Método de Euler:")
print(results_b)

plt.figure(figsize=(8, 5))
plt.plot(t_b, y_b, marker='o', linestyle='-', color='r')
plt.xlabel("t")
plt.ylabel("y")
plt.title("ODE")
plt.grid(True)
plt.show()
```

```
Resultados del Método de Euler:
      t         y
0   1.0  0.000000
```

```
1   1.2  0.200000
2   1.4  0.438889
3   1.6  0.721243
4   1.8  1.052038
5   2.0  1.437251
6   2.2  1.884261
7   2.4  2.402270
8   2.6  3.002837
9   2.8  3.700601
10  3.0  4.514277
```

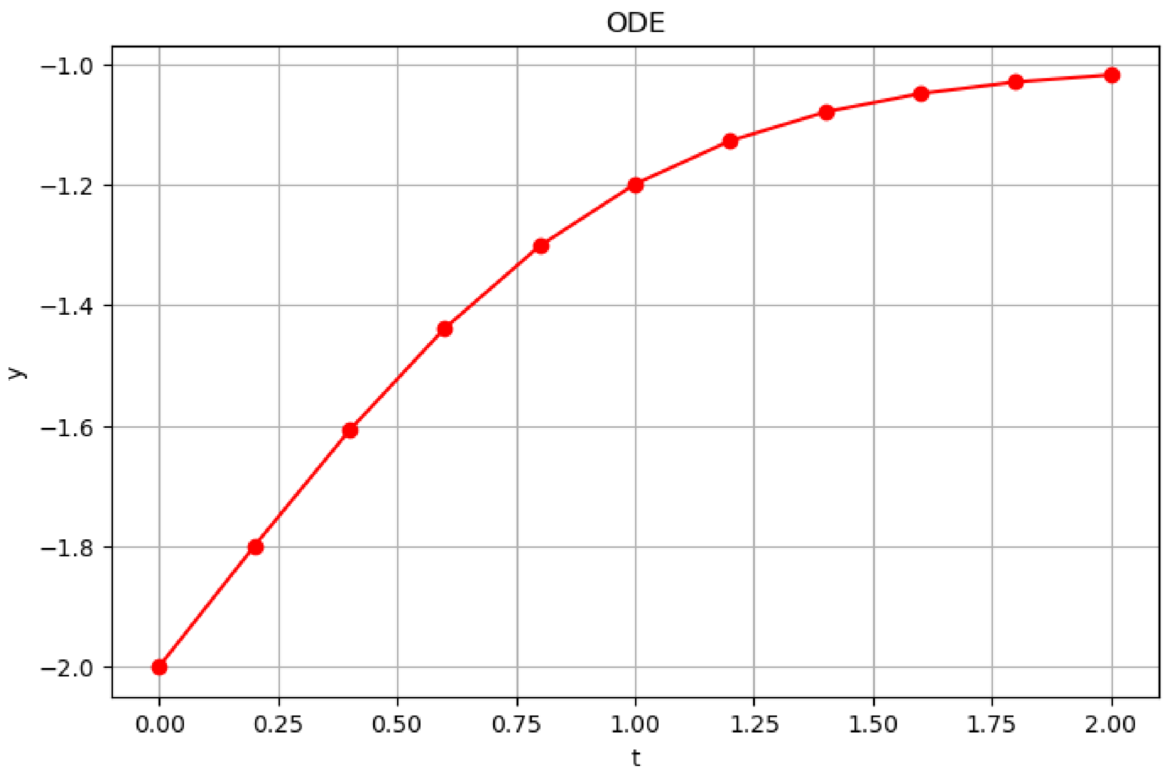


c. $y\acute{} = -(y+1)(y+3), 0 \leq t \leq 2, y(0) = -2$, con $h = 0.2$

```
f_c = lambda t, y: -(y + 1) * (y + 3)
t_c, y_c = euler_method(f_c, 0, -2, 0.2, 2)
results_c = pd.DataFrame({'t': t_c, 'y': y_c})
print("Resultados del Método de Euler:")
print(results_c)
```

```python
plt.figure(figsize=(8, 5))
plt.plot(t_c, y_c, marker='o', linestyle='-', color='r')
plt.xlabel("t")
plt.ylabel("y")
plt.title("ODE")
plt.grid(True)
plt.show()
```

```
Resultados del Método de Euler:
      t         y
0   0.0 -2.000000
1   0.2 -1.800000
2   0.4 -1.608000
3   0.6 -1.438733
4   0.8 -1.301737
5   1.0 -1.199251
6   1.2 -1.127491
7   1.4 -1.079745
8   1.6 -1.049119
9   1.8 -1.029954
10  2.0 -1.018152
```

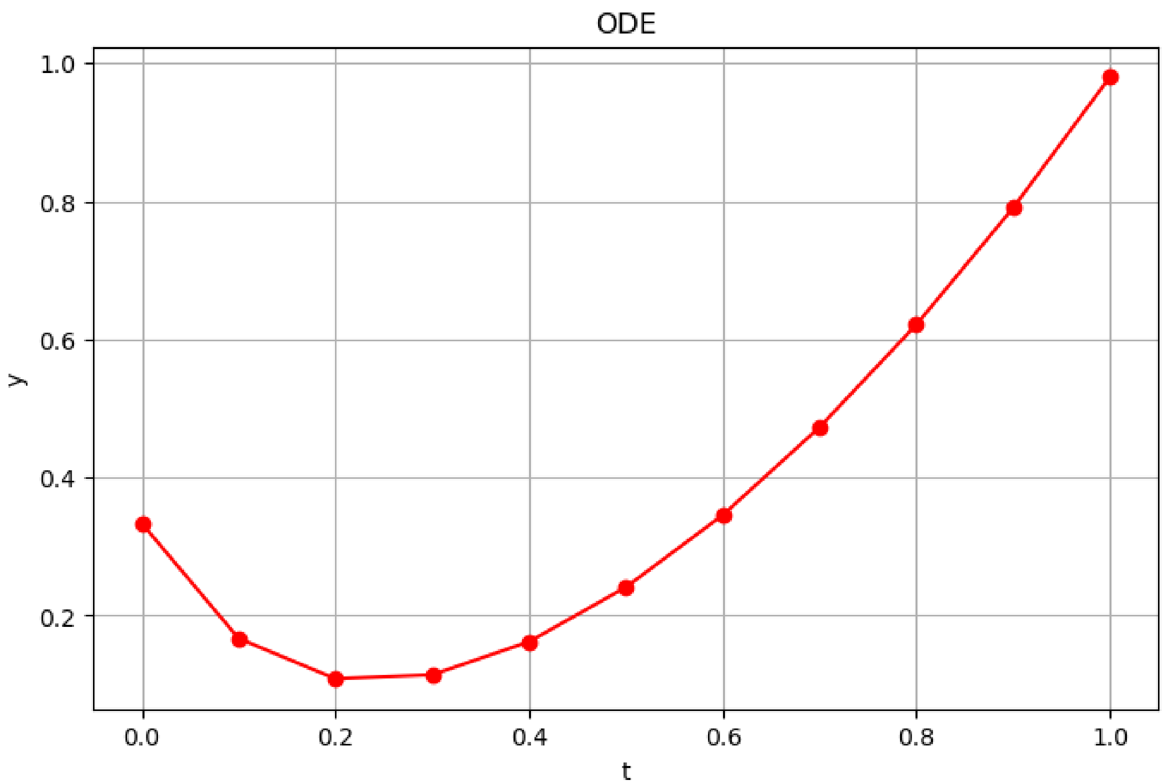


d. $y' = -5y + 5t^2 + 2t, 0 \leq t \leq 1, y(0) = 1/3$, con $h = 0.1$

```
f_d = lambda t, y: -5 * y + 5 * t**2 + 2 * t
t_d, y_d = euler_method(f_d, 0, 1/3, 0.1, 1)
results_d = pd.DataFrame({'t': t_d, 'y': y_d})
print("Resultados del Método de Euler:")
print(results_d)

plt.figure(figsize=(8, 5))
plt.plot(t_d, y_d, marker='o', linestyle='-', color='r')
plt.xlabel("t")
plt.ylabel("y")
plt.title("ODE")
plt.grid(True)
plt.show()
```

```
Resultados del Método de Euler:
      t         y
0   0.0  0.333333
```

```
1   0.1  0.166667
2   0.2  0.108333
3   0.3  0.114167
4   0.4  0.162083
5   0.5  0.241042
6   0.6  0.345521
7   0.7  0.472760
8   0.8  0.621380
9   0.9  0.790690
10  1.0  0.980345
```



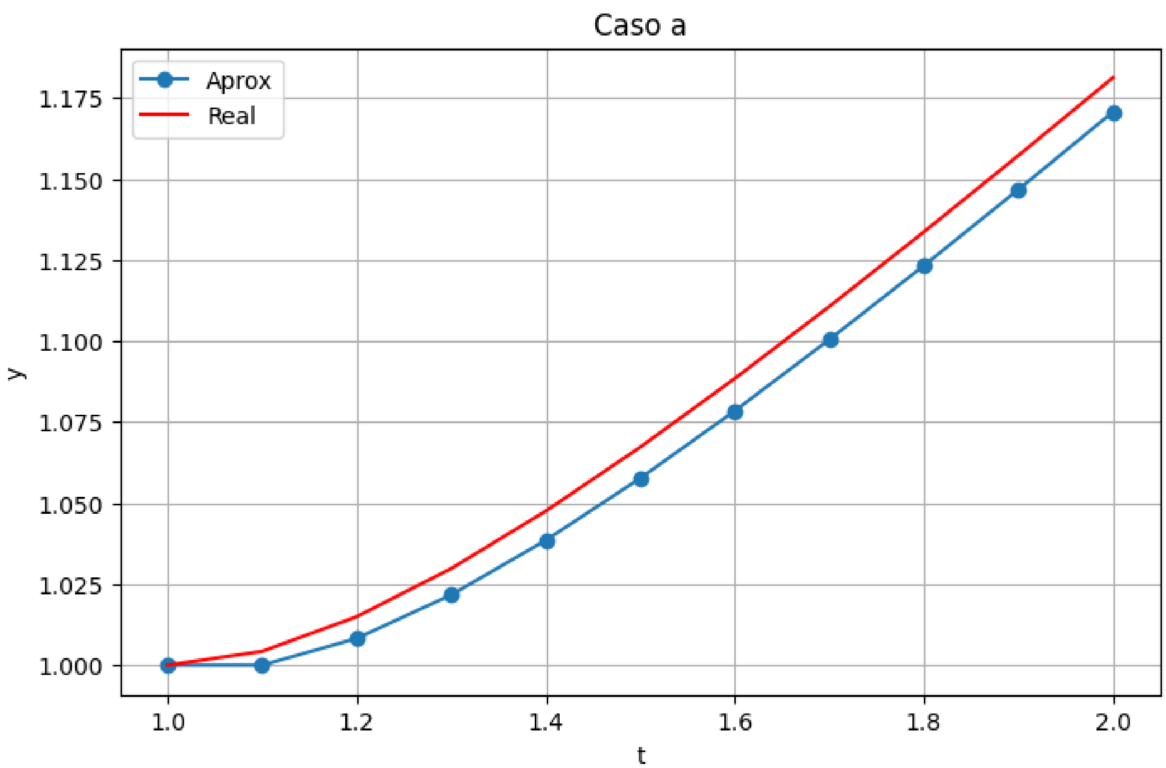
4. Aquí se dan las soluciones reales para los problemas de valor inicial en el ejercicio 3. Calcule el error real en las aproximaciones del ejercicio 3.

a. $y(t) = \frac{t}{1+ln(t)}$

```
f_a = lambda t, y: y / t - (y / t)**2
t_a, y_a = euler_method(f_a, 1, 1, 0.1, 2)
y_a_real = lambda t: t / (1 + np.log(t))
```

```python
error_a = np.abs(y_a_real(t_a) - y_a)
results_a = pd.DataFrame({'t': t_a, 'y_aprox': y_a, 'y_real': y_a_real(t_a), 'error': error_a

print("Errores del Método de Euler (Caso a):")
print(results_a)

plt.figure(figsize=(8, 5))
plt.plot(t_a, y_a, 'o-', label="Aprox")
plt.plot(t_a, y_a_real(t_a), 'r-', label="Real")
plt.title("Caso a")
plt.xlabel("t")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```

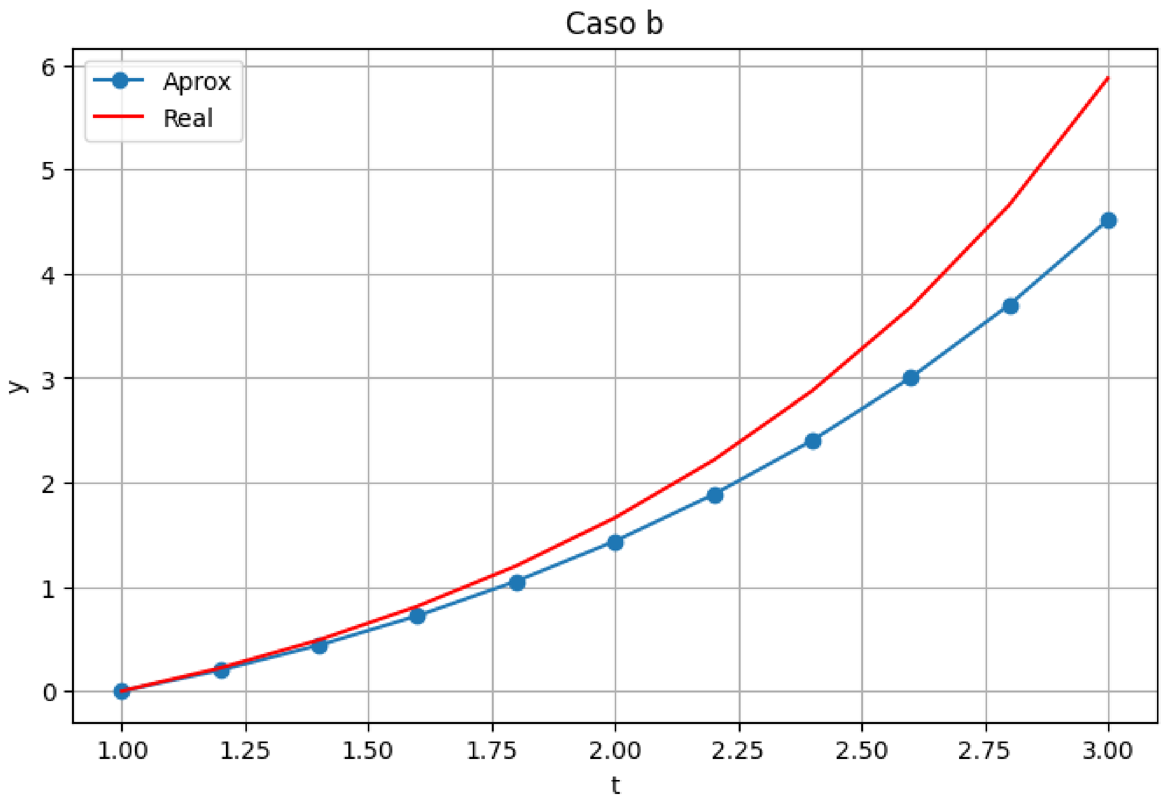
```
Errores del Método de Euler (Caso a):
      t   y_aprox    y_real     error
0   1.0  1.000000  1.000000  0.000000
1   1.1  1.000000  1.004282  0.004282
2   1.2  1.008264  1.014952  0.006688
3   1.3  1.021689  1.029814  0.008124
4   1.4  1.038515  1.047534  0.009019
5   1.5  1.057668  1.067262  0.009594
6   1.6  1.078461  1.088433  0.009972
7   1.7  1.100432  1.110655  0.010223
8   1.8  1.123262  1.133654  0.010392
9   1.9  1.146724  1.157228  0.010505
10  2.0  1.170652  1.181232  0.010581
```



b. $y(t) = t * tan(ln(t))$

```
f_b = lambda t, y: 1 + y / t + (y / t)**2
t_b, y_b = euler_method(f_b, 1, 0, 0.2, 3)
y_b_real = lambda t: t * np.tan(np.log(t))
error_b = np.abs(y_b_real(t_b) - y_b)
results_b = pd.DataFrame({'t': t_b, 'y_aprox': y_b, 'y_real': y_b_real(t_b), 'error': error_b

print("Errores del Método de Euler (Caso b):")
print(results_b)

plt.figure(figsize=(8, 5))
plt.plot(t_b, y_b, 'o-', label="Aprox")
plt.plot(t_b, y_b_real(t_b), 'r-', label="Real")
plt.title("Caso b")
plt.xlabel("t")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```

```
Errores del Método de Euler (Caso b):
      t    y_aprox    y_real     error
0    1.0  0.000000  0.000000  0.000000
1    1.2  0.200000  0.221243  0.021243
2    1.4  0.438889  0.489682  0.050793
3    1.6  0.721243  0.812753  0.091510
4    1.8  1.052038  1.199439  0.147401
5    2.0  1.437251  1.661282  0.224031
6    2.2  1.884261  2.213502  0.329241
7    2.4  2.402270  2.876551  0.474282
8    2.6  3.002837  3.678475  0.675638
9    2.8  3.700601  4.658665  0.958064
10   3.0  4.514277  5.874100  1.359823
```

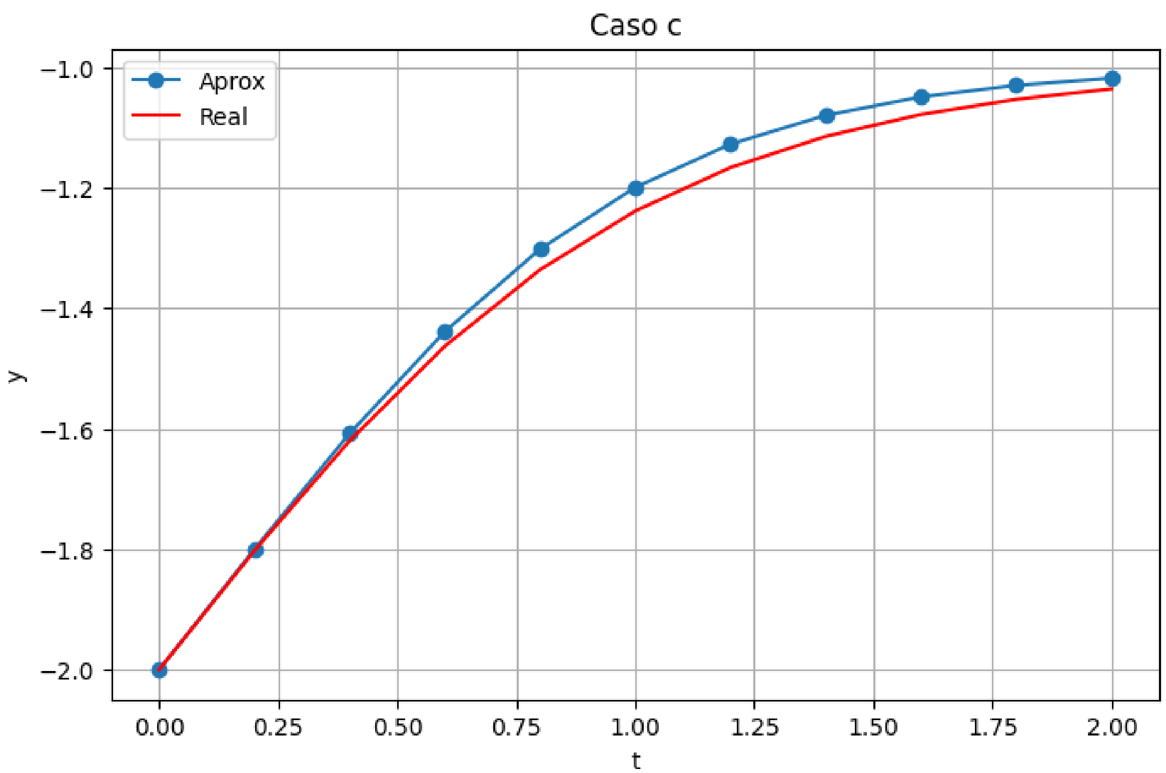


c. $y(t) = -3 + \frac{2}{1+e^{-2t}}$

```
f_c = lambda t, y: -(y + 1) * (y + 3)
t_c, y_c = euler_method(f_c, 0, -2, 0.2, 2)
y_c_real = lambda t: -3 + 2 / (1 + np.exp(-2 * t))
```

```python
error_c = np.abs(y_c_real(t_c) - y_c)
results_c = pd.DataFrame({'t': t_c, 'y_aprox': y_c, 'y_real': y_c_real(t_c), 'error': error_c

print("Errores del Método de Euler (Caso c):")
print(results_c)

plt.figure(figsize=(8, 5))
plt.plot(t_c, y_c, 'o-', label="Aprox")
plt.plot(t_c, y_c_real(t_c), 'r-', label="Real")
plt.title("Caso c")
plt.xlabel("t")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```

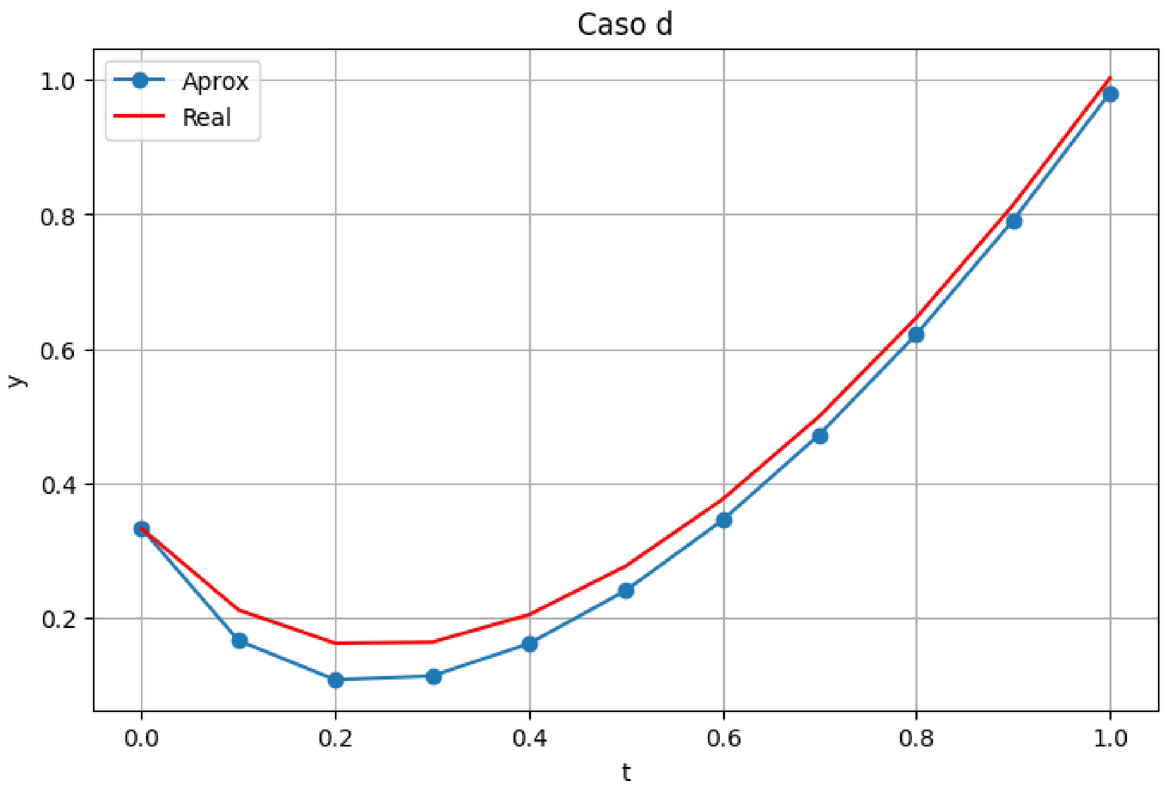
```
Errores del Método de Euler (Caso c):
      t   y_aprox    y_real     error
0   0.0 -2.000000 -2.000000  0.000000
1   0.2 -1.800000 -1.802625  0.002625
2   0.4 -1.608000 -1.620051  0.012051
3   0.6 -1.438733 -1.462950  0.024218
4   0.8 -1.301737 -1.335963  0.034226
5   1.0 -1.199251 -1.238406  0.039155
6   1.2 -1.127491 -1.166345  0.038854
7   1.4 -1.079745 -1.114648  0.034903
8   1.6 -1.049119 -1.078331  0.029212
9   1.8 -1.029954 -1.053194  0.023240
10  2.0 -1.018152 -1.035972  0.017821
```



d. $y(t) = t^2 + \frac{1}{3}e^{-5t}$

```
f_d = lambda t, y: -5 * y + 5 * t**2 + 2 * t
t_d, y_d = euler_method(f_d, 0, 1/3, 0.1, 1)
y_d_real = lambda t: t**2 + (1/3) * np.exp(-5 * t)
error_d = np.abs(y_d_real(t_d) - y_d)
results_d = pd.DataFrame({'t': t_d, 'y_aprox': y_d, 'y_real': y_d_real(t_d), 'error': error_c

print("Errores del Método de Euler (Caso d):")
print(results_d)

plt.figure(figsize=(8, 5))
plt.plot(t_d, y_d, 'o-', label="Aprox")
plt.plot(t_d, y_d_real(t_d), 'r-', label="Real")
plt.title("Caso d")
plt.xlabel("t")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```

```
Errores del Método de Euler (Caso d):
       t    y_aprox    y_real     error
0   0.0  0.333333  0.333333  0.000000
1   0.1  0.166667  0.212177  0.045510
2   0.2  0.108333  0.162626  0.054293
3   0.3  0.114167  0.164377  0.050210
4   0.4  0.162083  0.205112  0.043028
5   0.5  0.241042  0.277362  0.036320
6   0.6  0.345521  0.376596  0.031075
7   0.7  0.472760  0.500066  0.027305
8   0.8  0.621380  0.646105  0.024725
9   0.9  0.790690  0.813703  0.023013
10  1.0  0.980345  1.002246  0.021901
```



5. Utilice los resultados del ejercicio 3 y la interpolación lineal para aproximar los siguientes valores de ( ). Compare las aproximaciones asignadas para los valores reales obtenidos mediante las funciones determinadas en el ejercicio 4.

a. $y(0.25)$ y $y(0.93)$

```python
interp_a = interp1d(t_a, y_a, kind='linear', fill_value="extrapolate")
t_values_a = [0.25, 0.93]
y_aprox_a = interp_a(t_values_a)
y_real_a = [y_a_real(t) for t in t_values_a]
error_a = np.abs(np.array(y_real_a) - np.array(y_aprox_a))

for i, t in enumerate(t_values_a):
    print(f"t = {t}, y_aprox = {y_aprox_a[i]:.6f}, y_real = {y_real_a[i]:.6f}, error = {erro
```

```
t = 0.25, y_aprox = 1.000000, y_real = -0.647175, error = 1.647175
t = 0.93, y_aprox = 1.000000, y_real = 1.002772, error = 0.002772
```

b. $y(1.25)$ y $y(1.93)$

```python
interp_b = interp1d(t_b, y_b, kind='linear', fill_value="extrapolate")
t_values_b = [1.25, 1.93]
y_aprox_b = interp_b(t_values_b)
y_real_b = [y_b_real(t) for t in t_values_b]
error_b = np.abs(np.array(y_real_b) - np.array(y_aprox_b))

for i, t in enumerate(t_values_b):
    print(f"t = {t}, y_aprox = {y_aprox_b[i]:.6f}, y_real = {y_real_b[i]:.6f}, error = {erro
```

```
t = 1.25, y_aprox = 0.259722, y_real = 0.283653, error = 0.023931
t = 1.93, y_aprox = 1.302427, y_real = 1.490228, error = 0.187801
```

c. $y(2.10)$ y $y(2.75)$

```python
interp_c = interp1d(t_c, y_c, kind='linear', fill_value="extrapolate")
t_values_c = [2.10, 2.75]
y_aprox_c = interp_c(t_values_c)
y_real_c = [y_c_real(t) for t in t_values_c]
error_c = np.abs(np.array(y_real_c) - np.array(y_aprox_c))

for i, t in enumerate(t_values_c):
    print(f"t = {t}, y_aprox = {y_aprox_c[i]:.6f}, y_real = {y_real_c[i]:.6f}, error = {erro
```

```
t = 2.1, y_aprox = -1.012251, y_real = -1.029548, error = 0.017297
t = 2.75, y_aprox = -0.973894, y_real = -1.008140, error = 0.034246
```

d. $y(0.54)$ y $y(0.94)$

```
interp_d = interp1d(t_d, y_d, kind='linear', fill_value="extrapolate")
t_values_d = [0.54, 0.94]
y_aprox_d = interp_d(t_values_d)
y_real_d = [y_d_real(t) for t in t_values_d]
error_d = np.abs(np.array(y_real_d) - np.array(y_aprox_d))

for i, t in enumerate(t_values_d):
    print(f"t = {t}, y_aprox = {y_aprox_d[i]:.6f}, y_real = {y_real_d[i]:.6f}, error = {erro
```

```
t = 0.54, y_aprox = 0.282833, y_real = 0.314002, error = 0.031169
t = 0.94, y_aprox = 0.866552, y_real = 0.886632, error = 0.020080
```